

CLAIMS

1. A method for identifying ambiguities in a computer program, the method comprising:
 - automatically adding to the computer program, a temporary definition that comprises a name existing in a statement in the computer program;
 - wherein the computer program is expressed in a high level language that supports ambiguous usage of names as both function calls and memory accesses without an explicit indication therebetween;
 - automatically checking if the temporary definition reaches said statement along a first control flow path and at least one permanent definition of said name reaches said statement along a second control flow path;
 - wherein said at least one permanent definition pre-exists in said computer program prior to said automatically adding; and
 - automatically identifying said statement as containing a dual use of the name for each of (function call and memory access) if a result of said automatically checking is true.
2. The method of Claim 1 further comprising:
 - automatically checking if the temporary definition reaches said statement along each control flow path and if true automatically identifying said statement as containing a call to a function; and
 - automatically replacing said call to said function in said statement with a plurality of program instructions derived from a definition of said function.

3. The method of Claim 1 wherein further comprising:
automatically propagating said temporary definition along each control flow path that starts from said temporary definition in said computer program until said permanent definition or another permanent definition of said name is reached.
4. The method of Claim 3 wherein during said automatically propagating:
said temporary definition reaches said statement along a first control flow path; and
said temporary definition reaches said permanent definition in another statement along a second control flow path; and
wherein said another statement occurs in a sequence of statements between said temporary definition and said statement.
5. The method of Claim 3 wherein during said automatically propagating:
said temporary definition reaches said statement along a first control flow path;
said temporary definition reaches said permanent definition in another statement along a second control flow path; and
said permanent definition reaches said statement from said another statement along a continuation of the second control flow path.

6. The method of Claim 1 further comprising:
automatically constructing a structure of definitions and uses for a plurality of names in the computer program;
wherein said automatically constructing is performed after said automatically adding, so that the structure relates at least said temporary definition to said statement, and the structure relates every use of a name in said computer program to at least one definition; and said structure relates said temporary definition to said use of said name in said statement.
7. The method of Claim 6 further comprising, subsequent to said automatically constructing and prior to said automatically identifying:
automatically marking with a first flag each statement related by said structure to said temporary definition;
automatically with a second flag each statement related by said structure to said permanent definition; and
checking if said statement is marked with each of said first flag and said second flag.
8. The method of Claim 6 further comprising:
automatically using said structure to identify equivalence classes for said name; and
automatically renaming said name uniquely in each equivalence class.
9. The method of Claim 8 wherein:

said using and said renaming are performed subsequent to said constructing and prior to said identifying.

10. The method of Claim 8 wherein:

said using and said renaming are performed subsequent to said identifying.

11. The method of Claim 1 further comprising:

forming a plurality of flags comprising at least one set for each statement in the computer program, wherein flags in said set represent definitions of names in said computer program and for said statement a flag in the set when ON indicates that a definition represented by the flag reaches said statement and when OFF indicates that the definition represented by the flag does not reach said statement; and

wherein said automatically checking comprises checking if in a set of flags for said statement, a flag representing said temporary definition is ON and if another flag representing said permanent definition is also ON.

12. The method of Claim 1 further comprising:

forming a plurality of flags comprising at least three sets for each statement in the computer program, wherein flags in each set represent definitions of names in said computer program, said forming comprising:

turning on a first flag in a first set if the definition represented by the first flag occurs in said statement and turning off the first flag

OFF if the definition represented by the first flag does not occur in said statement;

turning on a second flag in a second set if the definition represented by the second flag is used in said statement and turning off the second flag if the definition is not used in said statement;

turning off a third flag in a third set if the definition represented by the third flag is not killed in said statement and turning on the third flag if the definition represented by the third flag is being killed by a definition in said statement; and

using said three sets of flags to form a fourth set of flags wherein a fourth flag when on indicates that a definition represented by the fourth flag reaches said statement and the fourth flag when off indicates that the definition represented by the fourth flag does not reach said statement.

13. The method of Claim 1 further comprising:

forming a plurality of tree nodes, said plurality of tree nodes comprising at least one node for each statement in the computer program;

wherein a tree node for said statement is automatically selected to be of a predetermined type indicating ambiguous usage if the name in said statement cannot be definitively determined from lexical analysis to be one of (memory access and function call);

forming a plurality of edges, each edge being connected between two nodes in said plurality of tree nodes, said plurality of edges comprising at least one edge for each control flow path reaching said statement;

automatically marking an edge into said tree node for said statement with a first flag if said statement is reachable by the temporary definition through said edge;

automatically marking the edge into said tree node with a second flag if said statement is reachable by said permanent definition through said edge;

repeating one of said automatically markings until all edges into said tree node are marked; and

checking if at least one edge into said tree node is marked with said first flag and if at least another edge into said tree node is marked with said second flag.

14. The method of Claim 13 further comprising:

replacing said tree node with another tree node if a result of said checking is false.

15. The method of Claim 1 further comprising:

automatically building a call graph for said computer program if no statement of said computer program contains dual use of any name.

16. The method of Claim 15 further comprising:

automatically replacing a call to a function in the call graph with a plurality of program instructions that perform said function, said plurality of program instructions being derived from a definition of said function and tailored to said call.

17. The method of Claim 15 further comprising:
performing iterative flow analysis on said call graph to determine propagation of constants.
18. The method of Claim 1 further comprising:
manually changing said computer program to eliminate the dual use in said statement, thereby to obtain a corrected version of said computer program; and
compiling said corrected version of said computer program without said temporary definition.
19. The method of Claim 18 further comprising, during said compiling:
automatically constructing another structure of definitions and uses that lacks said definition;
automatically using said another structure of definitions and uses to mark uses within said computer program as being function calls or memory accesses;
automatically constructing for said computer program a structure indicating the calling relationship among functions based on said marked uses; and
automatically using said structure indicating the calling relationship to improve said computer programs execution characteristics.
20. The method of Claim 1 wherein:

said automatically adding is performed for every name existing in said computer program.

21. The method of Claim 1 further comprising:
automatically performing lexical analysis on said computer program based on a grammar of said high level language;
automatically applying at least one predetermined rule, based on said lexical analysis, to check if said name can be definitively determined to be one of (memory access and function call); and
performing said automatically adding only if a result of said check is negative.

22. The method of Claim 21 wherein said predetermined rule comprises determining that said name is definitively a memory access by:
checking if at least said name followed by an open parenthesis is a target of an assignment on the left side of the “=” symbol in said statement.

23. The method of Claim 21 wherein said predetermined rule comprises determining that said name is definitively a function call by:
checking if said name is not explicitly defined in any statement of said computer program.

24. The method of Claim 21 wherein said predetermined rule comprises determining that said name is definitively a function call by:

checking if said name used as a multi-return function, invoked in said statement.

25. The method of Claim 1 wherein said automatically adding comprises:

inserting said temporary definition in an entry statement in source code of said computer program.

26. The method of Claim 1 wherein said automatically adding comprises:

inserting a temporary statement comprising said temporary definition, subsequent to an entry statement in source code of said computer program.

27. The method of Claim 1 wherein said automatically adding comprises:

modifying data structures used during construction of said plurality of nodes and edges such that said temporary definition appears to be in the program, without inserting a temporary statement comprising said temporary definition directly into said computer program or representation.

28. The method of Claim 1 further comprising, prior to any of (said automatically adding, said automatically constructing and said automatically identifying):

automatically translating said computer program into an intermediate representation modeled by a tree comprising a plurality of tree nodes and a plurality of tree edges; and

automatically marking a tree node for said statement as containing an ambiguous usage of said name, based on a grammar of said high level language.

29. The method of Claim 28 wherein said automatically adding comprises:

inserting in said tree a temporary node for said temporary definition;

wherein said temporary node is inserted at a location in said tree prior to said tree node and subsequent to an entry node of a function modeled by said tree.

30. The method of Claim 28 wherein said automatically adding comprises:

inserting said temporary definition to an entry node of a function modeled by said tree.

31. The method of Claim 28 wherein said automatically adding comprises:

inserting said temporary definition into a list of variables output by said statement;

wherein said list is associated with said tree node for said statement.

32. The method of Claim 1 wherein said automatically constructing comprises:

forming a plurality of nodes in said structure, one node for each statement in the computer program; and

adding a plurality of edges to said structure, such that each edge connects a first node, either directly or indirectly, to a second node if a definition in the first node can reach a use in the second node;

wherein said structure relates said temporary definition to said statement by an edge in said plurality of edges.

33. The method of Claim 1 further comprising,:

automatically performing an additional check to see if only the temporary definition reaches said statement along all control flow paths; and

automatically identifying said statement as containing a call to a function of said name if the result of said additional check is true;

automatically searching for said function with said name at all locations identified by at least one predetermined path; and

automatically flagging said name in said statement as being undefined if said function is not found during said searching.

34. The method of Claim 1 further comprising,

automatically performing an additional check to see if the temporary definition does not reach said statement along any control flow path; and :

automatically identifying said statement as containing a memory access of said name if the result of said additional check is true; and
automatically mapping said name to a specific location in physical memory of a size specified in said computer program.

35. The method of Claim 1 wherein:
said structure comprises a graph.

36. The method of Claim 1 wherein:
said structure comprises a plurality of bit vectors.

37. A method executed in a computer for at least partially resolving ambiguities in a computer program expressed in a high level language, the method comprising:

adding to said computer program, a definition comprising a name if said name is used ambiguously as both a function call and a memory access in a statement in said computer program;

constructing a graph of definitions and uses for a plurality of names including said name; and

identifying usage of said name in said statement to be definitively a memory access if in said graph all edges into said statement are from a pre-existing definition in said computer program.

38. A method executed in a computer for at least partially resolving ambiguities in a computer program expressed in a high level language, the method comprising:

- translating said computer program into an intermediate representation, wherein said intermediate representation models at least one statement in said computer program, said at least one statement comprising an ambiguous usage of a name as both a function call and a memory access;

- adding to an entry statement of said computer program, a definition comprising said name, said definition being marked artificial;

- constructing a graph of definitions and uses for a plurality of names including said name in said computer program, by adding a plurality of edges such that each edge connects a node in the intermediate representation, either directly or indirectly, to another node in the intermediate representation; and

- identifying usage of said name in said statement as a function call if in said definition-use graph all edges into said statement are from said artificial definition, and alternatively as a memory access if all edges into said statement are not from said artificial definition, and alternatively flagging said statement as a dual use.

39. A computer-readable storage medium encoded with a computer program and further comprising:

- a plurality of flags comprising at least four sets for each statement in the computer program, wherein flags in each set represent definitions of names in said computer program and for said statement:

a first flag in a first set when ON indicates that the definition represented by the first flag occurs in said statement and when OFF indicates that the definition represented by the first flag does not occur in said statement;

a second flag in a second set when ON indicates that the definition represented by the second flag is used in said statement and when OFF indicates that that the definition is not used in said statement;

a third flag in a third set when OFF indicates that the definition represented by the third flag is not killed in said statement and when ON indicates that the definition represented by the third flag is being killed by a definition in said statement;

a fourth flag in a fourth set when ON indicates that a definition represented by the fourth flag reaches said statement and when OFF indicates that the definition represented by the fourth flag does not reach said statement.

40. A computer-readable storage medium encoded with a computer program and further comprising a definition-use structure, said structure comprising:

a plurality of permanent nodes, one permanent node for each occurrence of a name in the computer program; and

a plurality of temporary nodes, wherein each temporary node is for a statement in the computer program that is of a predetermined type indicating ambiguous usage wherein the name in said statement cannot be definitively determined from lexical analysis to be one of (memory access and function call); and

a plurality of edges, wherein each edge connects a first node, either directly or indirectly, to a second node if a definition in the first node can reach a use in the second node, regardless of whether the first node is temporary or permanent.